

Curso de Python

1. Tipos de datos

En Python existen 4 tipos de datos y 4 estructuras de datos que detallaremos a continuación:

texto (str)	números	booleanos	
"Python" "750"	int 250 float 12.50	True False	
estructuras	mutable	ordenado	duplicados
listas []	✓	✓	✓
tuplas ()	✗	✓	✓
sets { }	✓	✗	✗
diccionarios { }	✓	✗*	✗:✓**

a. Dato tipo string

En Python el tipo de dato string es una cadena de caracteres, y para Python todo lo que está entre comillas es una cadena de caracteres. Las cadenas de caracteres se pueden concatenar con un signo **+**.

```
nombre="Luca"
```

```
apellido="Perez"
```

```
edad="55"
```

Ejercicio 5:

Realice un programa en el cual se le solicita el nombre y el apellido al usuario; y el programa devuelve una cadena de texto con las dos variables ingresadas.

```
Print ("Tu nombre es" + Input("Ingresa tu nombre:") + " " + Input("Ingresa tu apellido:"))
```

Ejercicio 6:

Realice un programa en el cual se le solicita el nombre y el apellido y la edad; y el programa le devuelve las 3 variables ingresadas concatenadas.

Ejercicio 7:

Realice un programa en el cual se le solicita su nombre, nro de celular y email; y el programa le devuelve las 3 variables ingresadas concatenadas.

b. Dato tipo Integer

En Python el tipo de dato Integer es un número entero.

```
num1=55
```

```
num2=11
```

Ejercicio 8:

Realice un programa en el cual se le solicita dos números al usuario y el programa devuelve la suma las dos variables ingresadas.

```
Print ( Input("Ingresa la variable1:") + Input("Ingresa la variable2:"))
```

c. Dato tipo float

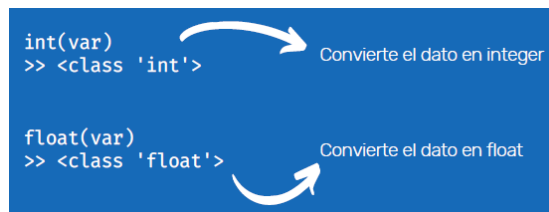
En Python el tipo de dato float es un número decimal.

```
num1=0.4
```

```
num2=11.55
```

i. Conversiones

Python realiza conversiones implícitas de tipos de datos automáticamente para operar con valores numéricos. En otros casos, necesitaremos generar una conversión de manera explícita.



ii. Redondear

En Python la función de redondeo nos permite redondear un número decimal y también le podemos indicar la cantidad de decimales que queremos trabajar.

d. Dato tipo Boolean

Los booleanos son tipos de datos binarios (**True/False**), que surgen de operaciones lógicas, o pueden declararse explícitamente.

e. Dato tipo List

Las listas son secuencias ordenadas de objetos. Estos objetos pueden ser datos de cualquier tipo: strings, integers, floats, booleanos, listas, entre otros. Son tipos de datos mutables.

mutable ✓ ordenado ✓ admite duplicados ✓

```
lista_1 = ["C", "C++", "Python", "Java"]
lista_2 = ["PHP", "SQL", "Visual Basic"]
```

indexado: podemos acceder a los elementos de una lista a través de sus índices [inicio:fin:paso]

```
print(lista_1[1:3])
>> ["C++", "Python"]
```

cantidad de elementos: a través de la propiedad len()

```
print(len(lista_1))
>> 4
```

concatenación: sumamos los elementos de varias listas con el símbolo +

```
print(lista_1 + lista_2)
>> ['C', 'C++', 'Python', 'Java', 'PHP', 'SQL', 'Visual Basic']
```

f. Dato tipo Diccionarios

Los diccionarios son estructuras de datos que almacenan información en pares **clave:valor**. Son especialmente útiles para guardar y recuperar información a partir de los nombres de sus claves (no utilizan índices).

mutable ✓ ordenado ✗ admite duplicados ✗

```
mi_diccionario = {"curso": "Python TOTAL", "clase": "Diccionarios"}
```

agregar nuevos datos, o modificarlos

```
mi_diccionario["recursos"] = ["notas", "ejercicios"]
```

acceso a valores a través del nombre de las claves

```
print(mi_diccionario["recursos"][1])
>> "ejercicios"
```

métodos para listar los nombres de las claves, valores, y pares clave:valor

keys() values() items()

g. Dato tipo Tuples

Los tuples o tuplas, son estructuras de datos que almacenan múltiples elementos en una única variable. Se caracterizan por ser ordenadas e inmutables. Esta característica las hace más eficientes en memoria y a prueba de daños.

mutable ✗ ordenado ✓ admite duplicados ✓

```
mi_tuple = (1, "dos", [3.33, "cuatro"], (5.0, 6))
```

indexado (acceso a datos)

```
print(mi_tuple[3][0])
>> 5.0
```

casting (conversión de tipos de datos)

```
lista_1 = list(mi_tuple)
print(lista_1)
>> [1, "dos", [3.33, "cuatro"], (5.0, 6)]
```

ahora es una estructura mutable

h. Dato tipo Set

Los sets son otro tipo de estructuras de datos. Se diferencian de listas, tuplas y diccionarios porque son una colección mutable de elementos inmutables, no ordenados y sin datos repetidos.

mutable ✓ ordenado ✗ ^{admite} duplicados ✗

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

métodos

add(item) agrega un elemento al set

```
mi_set_a.add(5)
print(mi_set_a)
>> {1, 2, "tres", 5}
```

clear() remueve todos los elementos de un set

```
mi_set_a.clear()
print(mi_set_a)
>> set()
```